Student-Centered Learning in a University Programming Course

Carlos Fontela, Pablo Suárez,
Fac. de Ingeniería Universidad de Buenos Aires, Argentina
{cfontela, psuarez}@fi.uba.ar

Abstract. In teaching Object Oriented Programming to university students, we have noticed that motivation is vital for the correct transmission of key concepts. In particular, engaging students provides a particular challenge that has changed over the recent years. To address this, in our OOP course in the University of Buenos Aires, we have been incorporating a variation of the flipped classroom approach we call Student Centered Learning. We have seen a good reception of this on behalf of our students, which has given us reason to maintain and improve on this strategy, while looking for more hard data so support the results.

Keywords: learner-centered teaching, problem-based learning, active learning, collaborative learning, object-oriented programming teaching.

1 Introduction

1.1 From teaching to learning

In the last few decades there has been a movement from a teacher-centered educational paradigm to an experienced-based learner-centered one. Definitely, we are shifting the focus from teaching to learning [1]. This movement is referred through different keywords, such as “active learning” [2], “collaborative learning” [3], “cooperative learning” [4], “constructivism”, “learner-centered” [5], “problem-based” [6], “student-centered learning” [7]. Bowman's book is perhaps the most disruptive from outside the world of academia beginning with its challenging title [8]. The goal of all these practices is to accomplish knowledge appropriation naturally through active exploration, switching the responsibility of learning to the students, and having them work in groups in problem-solving activities, while teachers concentrate on mediation and coaching. The focus is on students more than teachers and learning more than teaching. In this article, we will employ the expression Student-Centered Learning (SCL) to summarize all these practices.

We may think there is a revolution taking place in education. Nevertheless, learner-centered education has a long history, from Confucius and Socrates to Hegel and Pestalozzi. However, in the last fifteen years, research on learning, cognitive neuroscience, and other related subjects have given a scientific background to what

had been empirically observed. In addition, these studies stated many more findings, such as the importance of movement, multi-sensory stimuli, and conversation to keep the brain active to facilitate the appropriation of new knowledge.

In any case, since 2014, we have decided to try SCL in a programming course, where students could actually learn by doing, making mistakes with very little cost, and learning from their mistakes. The fact that the computer gives an immediate response to the students tries is another plus for the practice of SCL. All that said, we think these practices will help students learn more, and even make the activity of teaching more enjoyable, remembering we became teachers to improve students' lives. That year we explained our approach in a local workshop (WISIT 2014) and we gathered new recommendations and feedback [9].

1.2 Context

The course in which we work is the third programming course in the Software Engineering curricula, after one introduction to algorithms and programming with Python or C, and a data structures course. Therefore, the students reach our course with good background in basic programming. Our course is focused in Object Oriented Programming, with some topics of Object Oriented Design and Methodology.

The classes have a very large amount or students, with 150 to 200 in the classroom. The classroom is big, with steps that difficult both the students' and teachers' motion.

In this paper, we will describe how we manage to follow SCL practices, the limitations we are facing and some of the tradeoffs we have to make in our particular environment.

2 Related Works

There are many publications on SCL, its fundamentals, recommendations on applying and other aspects. An interesting book is the one from Doyle [10]. In this text Doyle makes interesting propositions, form SCL fundamentals to practical ways of classroom implementations.

Blumberg [11] offers us another interesting book that makes practical suggestions and tools to achieve SCL specifically in University courses.

Huba and Freed have undertaken the search for innovating in evaluation techniques for SCL educational environments [1]. There are other authors that have followed the same path, such as Saulnier et al. [12].

There are, however, very few, if any, publications which include actual classroom experiences, and even less in Latin America, nevertheless we know of many classes where SCL techniques are being used in teaching Computer Science and Software Engineering.

3 Learner-Centered Teaching in an Object Oriented Programming University Course

3.1 Classes

In the basic class structure we implement, in a course with 2 classes a week, each with a duration of 3 hours, we apply the following guidelines:

- No distinction between lectures and practice sessions.
- Many class start with a group dynamic as an opening activity (often involving students conversing in groups regarding a specific subject), as Bowman recommends [8].
- Two or three teachers per class.
- The students work with their computers in most classes.
- Ideally, two students share a computer doing pair programming.
- Sometimes, we start with a short lecture, involving as much interactive discussion with the students, and then we continue with exercises involving encapsulation, inheritance, polymorphism and some design aspects as well.
- Some classes revolve around the analysis of specific texts, having the students read them in the days prior to the class.
- Other class dynamics include starting with presenting the students with a practical exercise to resolve, and then presenting the theories behind the problem's resolution, introducing TDD, refactoring, polymorphism and other design principles.
- Role play is used for reinforcing certain more abstract subjects, such as delegation and MVC.
- Long lectures (over one hour) are avoided as a rule, as it is our perception that the current generation of students lose focus if practical exercises are not introduced frequently.
- Master classes? Yes, for specific subjects that will not be put into practice, but that require a conceptual explanation. In those cases, we make an effort to have as much interaction as possible with the students, asking questions every 15-20 minutes to reinforce what has been discussed, and trying to introduce real live anecdotes or experiences.

3.2 Practical Work

We have coursework that the students do outside the classroom.

According to Froyd y Simpson “the faculty member has the responsibility for forming teams” because “the teams that are formed influence the learning environment that is created” [7]. We have not observed the same, and believe in giving the students the liberty to build their own teams, reinforcing their involvement in the learning process.

- Groups of three or four students.

- In occasions, we evaluate the individual contribution of a specific student to their group.
- The feedback is given in-person.
- We make the students program games, since we believe this motivates them to look at the full scope of their work, and not just resolving a specific problem isolated from the real use of their work.
- The coursework is developed in an incremental fashion, following TDD guidelines, with feedback from a tutor in each incremental stage of their work.

3.3 Exams

In this course, according the current regulations, we have one mid-term exam and one final exam (called integration exam).

The mid-term exams are traditional tests, and in the following classes we resolve the test together with the students, and when the students get their grades, there is personalized feedback by with the teacher who graded the exam.

Final exams are done in the following manner:
- Students work on a computer (either their own, or one in our laboratories).
- They are given a set of questions and a short portion of working computer code, usually with important design flaws.
- Their task usually consists of identifying the code's shortcomings, suggesting improvements, expressing the improvements in UML diagrams and implementing them.
- They are usually interrupted by a teacher before finishing, and the written exam turns into an oral exam, where the student explain to the teacher his solution.
- The oral exam continues and the student is asked general theoretical questions spanning the semester's curricula.
- The student is given the result of the exam at the end of the oral examination.

We believe this form of examination is a better way to gauge the students knowledge, both giving them an opportunity to express themselves through their computer code, together with a more traditional oral examination.

3.4 Languages

The selection of computer languages is aligned with both the curricula and the particularities of the classroom dynamics.

We start with Smalltalk:
- No type checking.
- IDE is a part of the language platform.
- No configuration or deployment issues.

- This makes for a very contained environment, which helps students concentrate on the fundamental principles they are learning, with little distraction from non-OOP specific problems.

We continue with Java, which apart from being a strongly typed language, has a vast and readily available set of tools, including IDEs, code coverage, deployment frameworks, etc.

With these languages we focus on OOP principles, while in other courses after this one, the students work on more real-world and integrated assignments.

3.5 TDD

Test Driven Development (TDD) is used from the first class, initially using ad hoc tests and then incorporating a unit test framework (SUnit, JUnit).

This, along with refactoring, helps incorporating OOP concepts and understanding the benefits.

Towards the end of the course, reflection is introduced, and the use of frameworks like SUnit and JUnit helps understand the usefulness of reflection.

4 Discussion

4.1 Limitations

Are theoretical concepts actually understood?

Froyd and Simpson [7] found that many adopters indicate that this approach covers less content than traditional teaching methods, but we do not see any evidence of this happening.

Evaluating the students' advance is a challenge, since we consider short artificial exercises are do not challenge the students sufficiently.

We do not have classrooms with computers available to accommodate all our students, so we depend on students bringing their own equipment, and sharing.

The classrooms often do not have room to move around, and cannot change the arrangement of tables.

What if students are not inclined to collaborate? This teaching dynamic requires working in small groups, and some students prefer traditional approaches that require less involvement (sitting, listening, taking notes) and allow them to stay within their comfort zone.

This makes it vital for us to explain what we are doing and why, trying to convey to them that this approach is in their benefit.

Although we tailor the course as the weeks advance, according to the characteristics of the group, content coverage is high priority for faculty members.

4.2 Results

At the end of each course we have the students, voluntarily, fill out a survey to gauge their opinion, and we have a retrospective activity during the final class. Through these, the feedback we get shows a preference for more practical classes, group dynamics and the students involvement in the learning process.

These activities do not result in hard data. Objective numbers are difficult to obtain, in part due to how the students are evaluated (we do not have standardized tests, like multiple-choice exams, for example). We believe the evaluation is an integral part of the learning process, and standardized examinations do contribute little to this.

We perceive that students are progressively acquiring better critical-thinking skills and greater conceptual understanding, but have no hard data to support this.

5 Future improvements

In the next semesters, we are considering improvements along the following lines of work:

- Implementing actions to measure if the teaching method is really effective, since we are currently basing our decisions on opinions of teachers and students.
- Changes to the mid-term exam (which is mandatory), since currently it is a traditional written test.
- Explore ways to help students develop teamwork abilities.
- Measuring the rhythm of study, as is proposed by Fontdevila et al. [13].

7 Conclusions

We believe that students can be co-creators together with teachers, collaborating in the construction of knowledge. Fundamentally, we believe they need to be lifelong learners, in all aspects of knowledge and more so in a profession that will require them to learn continuously. In this sense, our approach attempts to favor independent learning, and we believe that in this way the skills and concepts will be better incorporated by our students.

Although we cannot realistically expect to completely change their learning habits in one course, we believe we can introduce them into new ways to approach the learning of new skills or tools.

Our work also leaves us with some important questions, one of the most outstanding is how to measure the benefits of the SCL approach we are implementing. This is one area we plan to work on in the next semesters.

References

1. Huba, M., Freed, J.: Learner-Centered Assessment on College Campuses: Shifting the Focus from Teaching to Learning. Journal Community College Journal of Research and Practice, vol. 24, Issue 9 (2000)
2. Bonwell, C. C., Eison, J. A.: Active Learning: Creating Excitement in the Classroom. George Washington University Press. (1991)
3. Bruffee, K. A.: Collaborative Learning and the "Conversation of Mankind". College English, 46(7), pp. 635-652. (1984)
4. Johnston, S., Cooper, J.: Quick-thinks: Active-thinking Tasks in Lecture Classes and Televised Instruction. Cooperative Learning and College Teaching, 8(1), pp. 2-7. (1997)
5. Bilimoria, D., Wheeler, J.V.: Learning-centered education: a guide to resources and implementation. Journal of Management Education, 29(3), pp. 402-428. (1995)
6. Blumberg, P.: Problem-based learning: a prototypical example of learning-centered teaching. Journal of Student Centered Learning, 3(2), pp.111-125. (2007)
7. Froyd, J., Simpson, N.: Student-Centered Learning. Addressing Faculty Questions about Student-Centered Learning. What is meant by Student-centered Learning (SCL). Science Education (1997)
8. Bowman, S. L.: Training from the back of the room!: 65 ways to Step aside and let them learn. John Wiley & Sons. (2008)
9. Fontela, C., Páez, N., Suárez, J.P.: Lecciones aprendidas enseñando OO en UBA Ingeniería. Workshop de Ingeniería en Sistemas y Tecnologías de Información. (2014)
10. Doyle, T.: Learner-centered teaching: Putting the research on learning into practice. Stylus Publishing, LLC. (2012)
11. Blumberg, P: Developing Learner-Centered Teaching: A Practical Guide for Faculty. Jossey-Bass. (2008)
12. Saulnier, B., Landry, J.; Longenecker, H. Jr; Wagner, T.: From Teaching to Learning: Learner-Centered Teaching and Assessment in Information Systems Education. Journal of Information Systems Education; West Lafayette 19.2, pp. 169-174 (Summer 2008)
13. Fontdevila, D., Tugnarelli, M., Ismael, S., & Videla, L.: Promoción del ritmo de estudio por feedback colectivo de progreso en trabajos prácticos. In XXI Congreso Argentino de Ciencias de la Computación CACIC. (2015)
6. National Center for Biotechnology Information, http://www.ncbi.nlm.nih.gov